24.1 Circuit Challenges from Cryptography

Ingrid Verbauwhede, Josep Balasch, Sujoy Sinha Roy, Anthony Van Herrewege

KU Leuven, Heverlee, Belgium

Implementing cryptography and security into integrated circuits is somehow similar to applications in other fields. We have to worry about comparable optimization goals: area, power, energy, throughput and/or latency. Moore's law helps to attain these goals. However, it also gives the attackers more computational power to break cryptographic algorithms. On top of this, quantum computers may become soon a reality, so that novel, very computationally demanding "post-quantum" cryptographic algorithms need implementation. Finally, there is a third dimension to the problem: implementations have to be resistant against physical attacks and countermeasures increase the cost. This paper demonstrates with actual data how these conflicting challenges are being addressed.

More Moore: The strength of an algorithm is expressed by the computational complexity of known cryptanalytic tools to 'break' the algorithm on classical computers. The computational complexity for secret key algorithms such as DES or AES is of the form 2^b with b the key length: i.e. the effort of trying all possible keys. b equal to at least 128 is recommended for secret key algorithms. Public key algorithms are based on mathematical problems which require much larger key lengths: a key length of at least 3284 bits is recommended for RSA, while 256 bit keys are the norm for elliptic curve-based cryptosystems (ECC). Fig 1(top) shows the minimum key lengths of various algorithms for cryptographic security for the next 40 years and beyond, the consequence of Moore's law.

Symmetric key algorithms, such as AES, use arithmetic that runs inefficiently on standard instruction sets or C compilers. Energy efficiency, measured as Gbits/Joule, is shown on Fig 1 (bottom) for the two most popular modes of operation: the Counter Mode (CTR), which can be pipelined, and the Cipher Block Chaining (CBC) encryption mode, which puts the AES engine in a feedback mode, which cannot be pipelined. It illustrates how dedicated ASICs or instructions are more energy efficient than generic FPGA or SW implementations [1][2][3]. It also shows the improvement over technology nodes. E.g. AES instructions on the Haswell Intel platform require 0.63 and 4.44 cycles/byte for CTR and CBC-encryption respectively [1], which converted to Gbits/Joule is many orders of magnitude more energy efficient than generic assembly or C-compiled code.

Physical security: As the computational complexity of the algorithms improved and mathematical attacks became practically infeasible, the focus of attackers moved towards the physical implementation. Physical attacks are broadly split into two main classes. Passive side-channel attacks observe data-dependent variations in timing, power consumption and/or electromagnetic radiation of cryptographic algorithms during calculations, from which the key or other sensitive data is revealed. Active attacks intentionally introduce faults by manipulating clock or power supply or introduce glitches. More advanced attackers shoot with lasers or perform EM manipulations. Unfortunately, many countermeasures are expensive and conflict. Countermeasures against side-channel attacks might make fault-attacks easier. At this moment, practical schemes provide resistance but not provable resistance, while mathematically provable secure systems (against a well defined attacker model) remain unrealistic in practice. Wave Dynamic Differential Logic (WDDL) was the first practical circuit level countermeasure demonstrated on the size of a complete AES co-processor to resist Differential Power and Electromagnetic Attacks [3]. The idea is to make the power consumption profile independent of the actual data being processed by switching exactly once the same amount of energy every clock cycle. A WDDL exor/nexor gate is shown on Fig 2. For circuit level countermeasures, the quality of the resistance depends on the quality of the circuit design including balanced place and route. Mathematical countermeasures such as threshold implementations aim at being independent of the actual circuit implementation [4]. They split the data in different 'shares' and make sure that at no point in time power consumption information is available from all shares, see Fig 2. However in practice, heavy pipelining is required to keep the shares apart [4] and temporal or spatial separation is difficult to enforce. It also requires access to a large amount of randomness.

Post-quantum security: The security of current mainstream public key algorithms RSA and DSA, ECC, and ECDSA (EC Digital Signature Algorithm) is based on the computational effort to break mathematically 'hard' problems such as factorization or the discrete log. Unfortunately, it was shown by Shor that these problems become 'easy' to solve in case quantum computers become a reality. Hence, the cryptographic community is working on algorithms, which remain secure in the quantum computer era. So far, most of them have a much higher computational complexity. One novel approach is the lattice based Ring Learning with Errors (RLWE) crypto system [5]. In this scheme, messages are encoded and represented by finite field polynomials, typically with 256 or 512 coefficients, each coefficient of size 13 to 14 bits (in NTT format). The encryption process consists of two major steps: first creating error polynomials, which involves generating random numbers and using them in a discrete Gaussian sampler. Secondly, they are multiplied and added with the message polynomial. Straightforward schoolbook multiplication of two polynomials, requires $O(n^2)$ multiplications. For large n, in this case 256 or 512, conversion to the Number Theoretic Transform (NTT) domain reduces this to O(nlogn) number of multiplications. Three different implementations of the RLWE processor are made in ASIC (synthesized results), FPGA and SW running on an ARMCortex-M4F, (Fig. 3). Fig. 4 shows the architecture of a compact RLWE co-processor, with the best area-time performance published so far [5]. With progress on the theoretical side, more efficient fully-homomorphic schemes based on the RLWE problem are appearing. So far, for 80 bit security (so less than AES), it requires a polynomial of degree n= 32768 and field size 1225 bits, which determines the number of multiplications and the size of the multiplier. Thus the challenge for the circuit designer is to realize these huge multipliers and to provide sufficient entropy for the random number generation.

PUFs: In the future, cryptography and hardware will be even more entangled, making active use of features provided by silicon technology. Cryptography by itself does not provide security: it needs securely stored secret keys, high quality random numbers, nonces (non repeating numbers), unforgeable unique IDs and more. It relies for that on circuits and silicon technologies which are poorly characterized for security purposes. PUFs, Physically Uncloneable Functions, are a first enabling technology for this, producing uncloneable and inherent instance-specific measurements of integrated circuits, comparable to human fingerprints. PUF behavior is the result of processing variations and circuit design decisions. E.g. the start-up value of a 6-TOR SRAM cell is a source of PUF material, if it is nicely differentially designed and laid out. We derived experimentally, using a black-box approach without knowing the internal organization of the SRAM, the amount of PUF material available in the SRAM of commodity micro-controllers. Robustness (= same value at each start-up) and uniqueness (= difference between any two samples of same device) are measured with Hamming weight, within-class-Hamming-distance, and between-class-Hamming-distance, which should reach 50%, 0%, and 50% respectively. Results are shown in Figs 5 and 6 for the STMicroelectronics STM32F100R8 and the Microchip PIC16F1825 [6]. The STM32F100R8 has very good PUF behavior while the PIC16F1825 behaves poorly. PUF behavior and noise for random number generations are two ends of the same scale. For PUFs we prefer a large variability (i.e. random process variation but no bias), stable over multiple measurements with small effects of environmental noise, while if we want to extract min-entropy for a RNG we expect circuits that are sensitive to environmental noise. We use the unstable startup bits of the SRAMs as a seed for a random number generator [6].

Acknowledgements: this work is partially supported by IMinds, FWO, and the EU under FP7 programs Puffin and HINT, and an Erasmus Mundus fellowship.

References:

[1] http://ark.intel.com/products/80807 Shay Gueron, Intel, and Chris Nicol, WaveSemi, private communication.

[2] S. Mathew et al, "340mV–1.1V, 289Gbps/W, 2090-gate NanoAES HW accelerator with area-optimized encrypt/decrypt GF(2^4)^2 polynomials in 22nm tri-gate CMOS," IEEE Symp on VLSI circuits, 2014.

[3] D. Hwang et al., "AES-Based Security Coprocessor IC in 0.18-um CMOS with Resistance to Differential Power Analysis Side-Channel Attacks," IEEE JSSC 41(4), pp. 781-792, 2006.

[4] A. Moradi et al., "Pushing the Limits: A Very Compact and a Threshold Implementation of AES, " EUROCRYPT 2011, LNCS 6632, 2011, pp 69-88.

[5] Sujoy S. Roy et al., "Compact Ring-LWE cryptoprocessor," CHES, 2014.

[6] A. Van Herrewege et al., "Secure PRNG Seeding on Commercial Off-the-Shelf Microcontrollers," IEEE Conf. on Trustworthy Embedded Devices, 2013.

Protection	Symmetric	RSA	ECC	Ring-LWE	Hash
2014 – 2040	128	3 284	256	3 328	256
> 2040	256	15 424	512	7 168	512

†Based on "ECRYPT II Yearly Report on Algorithms and Key Sizes" (2012).

AES performance on various platforms	Throughput [Gbit/s]	Power [W]	Figure of merit vs *best*
22 nm CMOS (AES128) [4]	0.086	0.46×10^{-3}	1
65 nm CMOS (AES256) [1]	9.100	0.10	1/2
FPGA (Xilinx Kintex-7) (AES256) [1]	3.908	0.20	1/10
180 nm CMOS (CTR or CBC) [3]	4.123	0.35	1/16
Intel 4790K (Haswell) (CTR) [1]	50.799	88	1/322
(CBC) [1]	7.208	88	1/2270
Qualcomm Krait (AES256) [1]	0.228	1	1/817
Intel Core2 Quad Q9950 (CTR, bitsliced, ASM)	3.275	95	1/5396
ARM Cortex-M4F (C, gcc 4.8.4)	0.536×10^{-3}	0.02	1/7711

Figure 24.1.1: Recommended key length; AES on various platforms and technologies

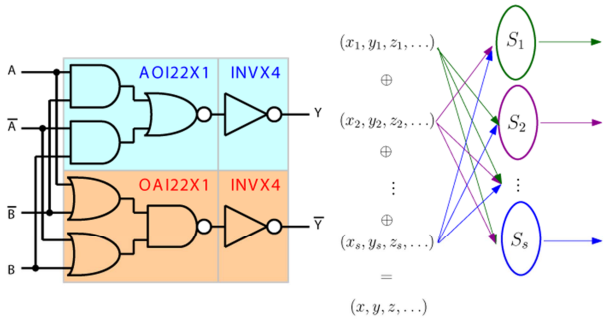

Figure 24.1.2: Wave Dynamic Differential Logic and Threshold countermeasures

	ASIC 130 nm CMOS	FPGA Xilinx Virtex-5	Software ARM Cortex-M4F
Cycles			
Sampling[1]	269	269	7.2K
Fwd NTT	1.7K	1.7K	30.5K
Inv NTT	2.3K	2.3K	37.9K
Enc	6.3K	6.3K	122K
Dec	2.8K	2.8K	42.7K
Freq. [MHz]	500	313	168
Time [µs]			
Enc	12.6	20.1	726
Dec	5.7	9.1	254
Area	11.5K GE[2]	1349/860/1 (LUT/FF/DSP)	1.5 KiB/5.8 KiB (Flash/RAM)

[1]Cycles to sample 256 error-poly. coef. Avg. case: 8.2 random bits / coef.
[2]Area does not include RNGs. HW requires six 256 × 26 bit RAMs.

Figure 24.1.3: Ring LWE post-quantum crypto on ASIC, FPGA and ARM Cortex-M4F SW

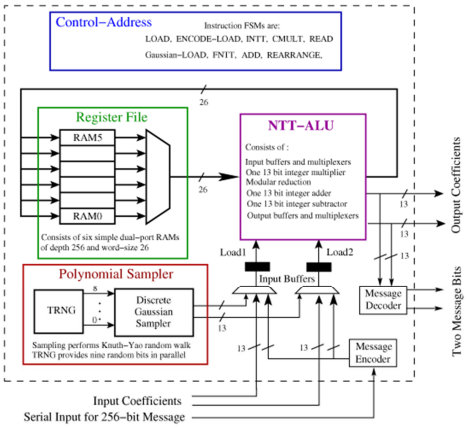

Figure 24.1.4: Compact Ring LWE architecture

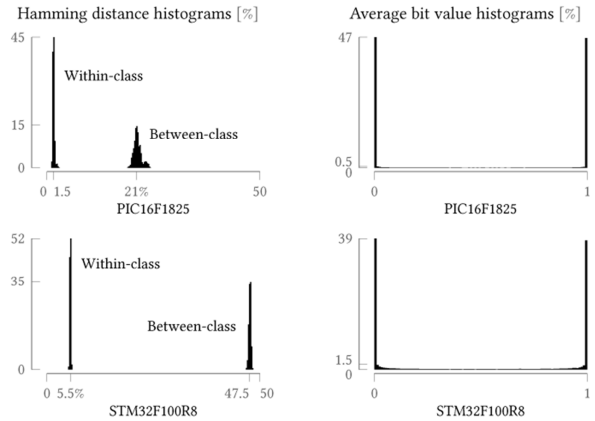

Figure 24.1.5: PUF behavior of SRAMs on commodity micro-controllers

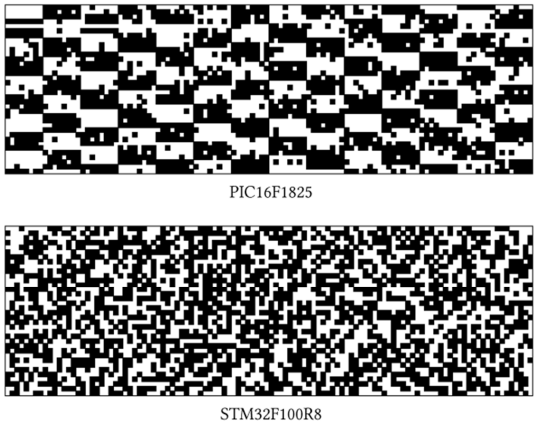

Figure 24.1.6: Visual representation of SRAM PUF responses ('0' is white, '1' is black)